第二十三节 服务治理（下）

Dubbo Filter

API 定义

```
@SPI
public interface Filter {
    /**
     * Make sure call invoker.invoke() in your
implementation.
     */
    Result invoke(Invoker<?> invoker, Invocation
invocation) throws RpcException;

    interface Listener {

        void onResponse(Result appResponse,
Invoker<?> invoker, Invocation invocation);

        void onError(Throwable t, Invoker<?>
invoker, Invocation invocation);
    }
```

```
}
```

```
@SPI
public interface Filter {
    /**
     * Make sure call invoker.invoke() in your
implementation.
     */
    Result invoke(Invoker<?> invoker, Invocation
invocation) throws RpcException;

    boolean isUsedForProvider() default true;

    boolean isUsedForConsumer() default true;
}
```

内建实现

org.apache.dubbo.rpc.filter.AccessLogFilter

设计缺陷

org.apache.dubbo.rpc.Filter 与 Filter Chain 之间的关系对于开发人员而言，不明确。

比如 Servlet Filter

Filter 和 FilterChain，FilterChain 控制 Filter 是否继续执行下一个。

注意事项

Filter 实现需要指明 `@Activate` 来确定是否适用于服务提供方（Provider）和服务消费方（Consumer）

实现细节

服务提供方

```
<T> Exporter<T> export(Invoker<T> invoker) throws RpcException;
```

```
    public <T> Exporter<T> export(Invoker<T>
invoker) throws RpcException {
        if (UrlUtils.isRegistry(invoker.getUrl()))
{
            return protocol.export(invoker);
        }
        return
protocol.export(buildInvokerChain(invoker,
SERVICE_FILTER_KEY, CommonConstants.PROVIDER));
    }
```

```
?service.filter=filter1&service.filter=filter2
```

```
String serverFilter =
url.getParameter("service.filter"); //
filter1,filter2
```

服务消费方

```
<T> Invoker<T> refer(Class<T> type, URL url)
throws RpcException;
```

```
    @Override
    public <T> Invoker<T> refer(Class<T> type, URL
url) throws RpcException {
        if (UrlUtils.isRegistry(url)) {
            return protocol.refer(type, url);
        }
        return
buildInvokerChain(protocol.refer(type, url),
REFERENCE_FILTER_KEY, CommonConstants.CONSUMER);
    }
```

设计模式

责任链模式

拦截模式

前置拦截

```
public boolean preFilter(Invoker<?> invoker,
Invocation invocation);
```

后置拦截

```
public void postFilter(Invoker<?> invoker,
Invocation invocation,Result result);

public Result postFilter(Invoker<?> invoker,
Invocation invocation,Result result);
```

异常拦截

```
public void onError(Invoker<?> invoker, Invocation
invocation,Throwable cause);

public Result onError(Invoker<?> invoker,
Invocation invocation,Throwable cause);
```

Dubbo Logger

类似于 Java Commons Logging 或 slf4j